

Név: EHA: Neptun (ha tudod):

Gyakvez: Farkas Richárd ☐ Gelle Kitti ☒ Gosztolya Gábor ☐ Kövesi-Nagy Dániel ☐

Sliz-Nagy Alex ☐ Szántó Zsolt ☐

1. Feladat, 6 pont. Adott egy építőkocka-készlet. Van köztük 1 egység magas, van 2 egység magas kék, van 2 egység magas piros és van 3 egység magas is. (Mind a négy fajta kockából tetszőlegesen sok áll rendelkezésre). Írj (pszeudo)kódot egy olyan dinamikus programozási algoritmussal, amely kiszámolja, hogy hányféleképpen építhető fel egy N magasságú torony.

```
int HanyfeleTorony(int N) {
    int[] tomb = new int[N+1];
    tomb[1] = 1;
    tomb[2] = 3;
    tomb[3] = 6;
    for (int i=4; i<=N, i++) {
        tomb[i] = tomb[i-1] + tomb[i-2] + tomb[i-2] + tomb[i-3];
    }
    return tomb[N];
}
```

2. Feladat, 6 pont. *Csak egyszer előforduló szavak.* A bemenetünk egy tömb, melynek minden eleme egy szó. Írj egy algoritmust, mely maximum $O(n \cdot \log n)$ időben meghatározza, mely szavak fordulnak elő pontosan egyszer a szövegben!

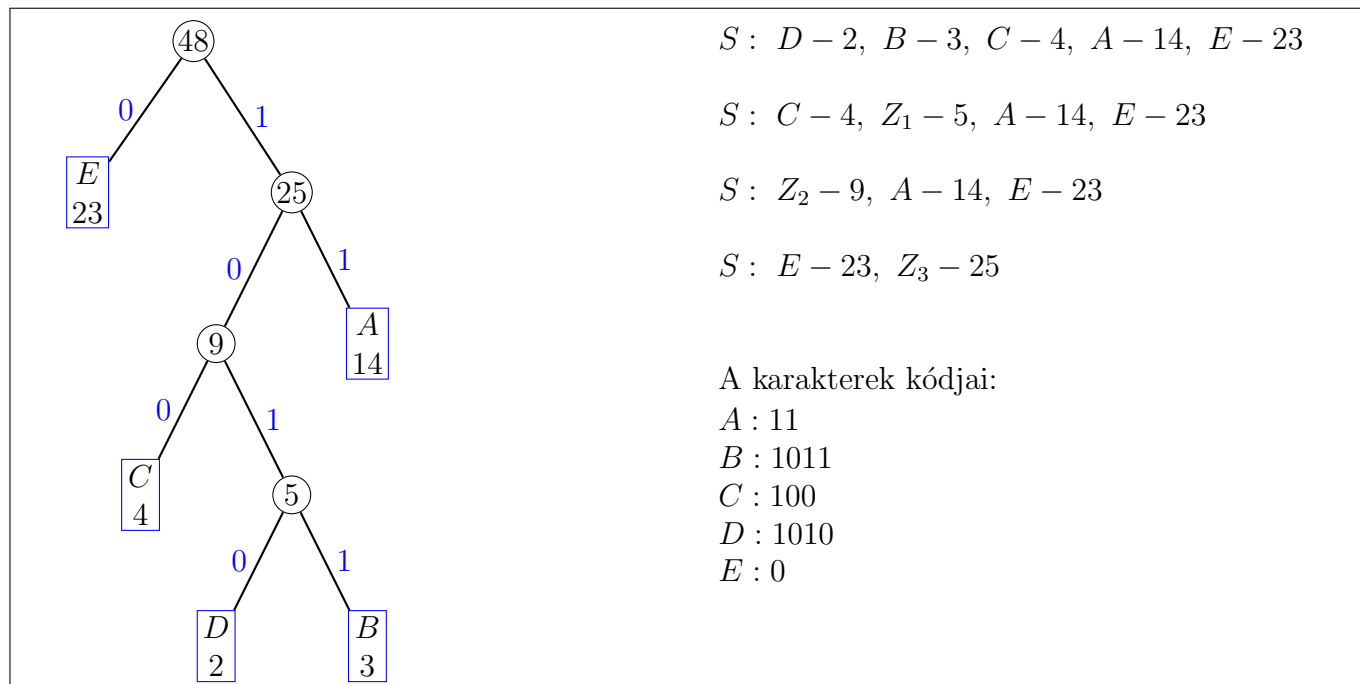
```
freqs1(words)
    freqs = map()           //Szo - gyakorisag parosokat tartalmazo map
    result = list()

    for word in words
        freqs[word]++

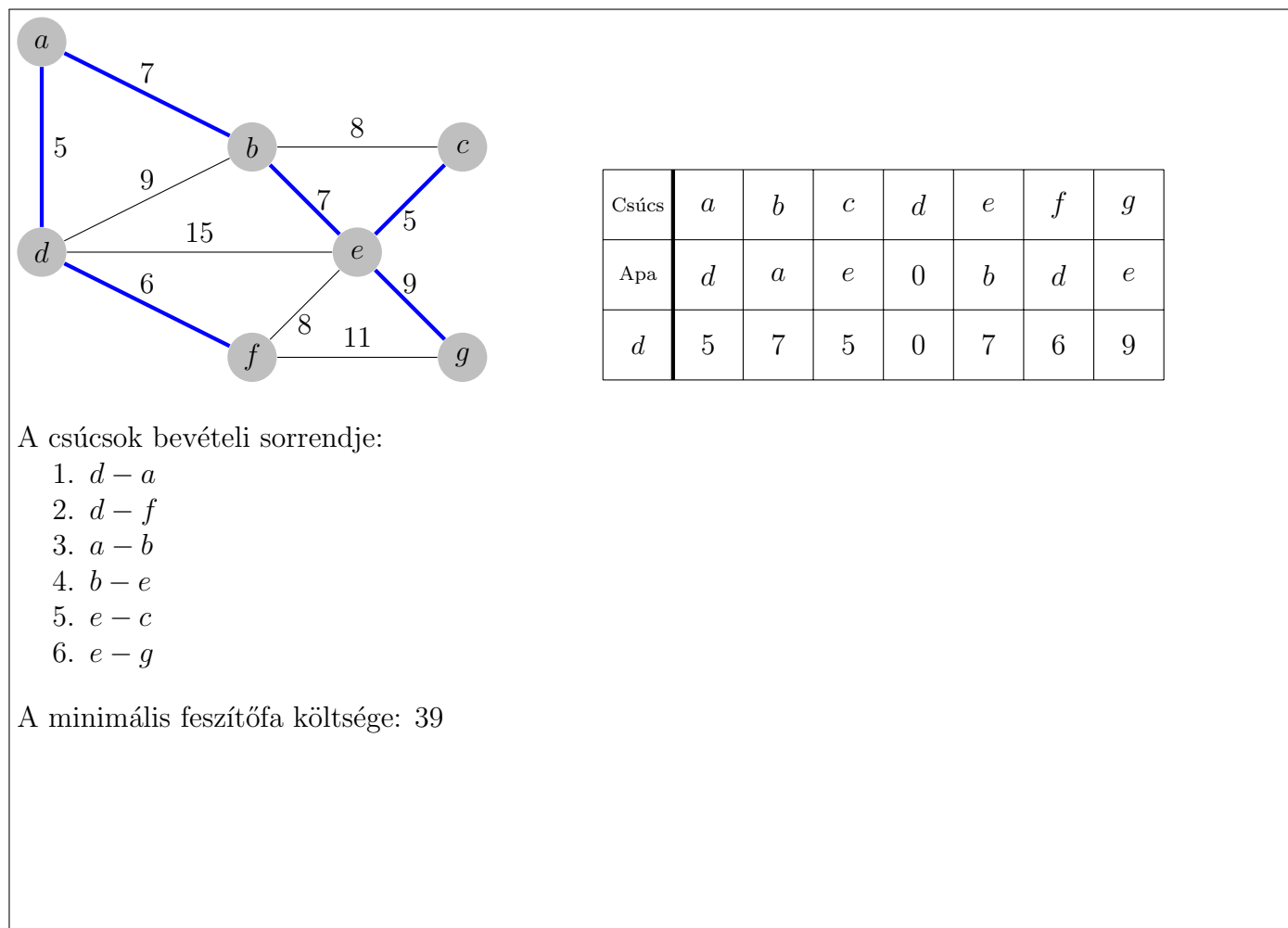
    for word in freqs       //A freqs map szavain iterálunk végig
        if freqs[word] == 1
            result.add(word)

    return result
```

3. **Feladat, 6 pont.** Mi az optimális Huffman kódja az alábbi abc-nek: $f(A) = 14$, $f(B) = 3$, $f(C) = 4$, $f(D) = 2$, $f(e) = 23$? A csúcsokba kerüljenek be a rájuk vonatkozó gyakorisági értékek!



4. **Feladat, 6 pont.** Hajtsd végre a Prim algoritmust a következő gráfon a d csúsból indulva! Jelöld az élek bevételi sorrendjét, vagy töltsd ki a táblázatot! Mennyi a minimális feszítőfa költsége?



5. Feladat, 5 pont. Adott a következő kódrészlet. Magyarázd el, mit csinál a kód!

```
class ModositottVerem {
    int N;
    int x;
    int y;
    int [] tomb;

    public ModositottVerem (int N) {
        this.N = N;
        this.x = 0;
        this.y = N - 1;
        this.tomb = new int [N];
    }

    public int push_small(int elem){
        if (this.x<=this.y) {
            tomb[x] = elem;
            x++;
            return x-1;
        }
        return -1;
    }
}
```

```
public int push_big(int elem){
    if (this.x<=this.y) {
        tomb[y] = elem;
        y--;
        return y+1;
    }
    return -1;
}

public int pop_small(){
    if (x > 0) {
        x--;
        return tomb[x];
    }
    return -1;
}

public int pop_big(){
    if (y < N) {
        y++;
        return tomb[y];
    }
    return -1;
}
}
```

Egy olyan veremszerű adatszerkezetet valósít meg, amely két (pozitív egészeket tartalmazó) vermet egy $N + 1$ méretű közös t tömbben tárol „egymás felé fordítva”, azaz a tömb két végéről indulnak és két külön változóban tárolja a vermek pillanatnyi tetejének indexét.

- **Push_small(x)**: ha fér még elem a tömbbe, akkor x -et a kisebb indexeken lévő verembe teszi és visszatér az index régi értékével (azaz ahova rakta). Ha nem fér be, akkor nem teszi be és -1 -gyel tér vissza.
- **Push_big(x)**: ugyanez a nagyobb indexeken lévő veremre vonatkoztatva.
- **Pop_small()**: ha van elem a kisebb indexeken lévő veremben, akkor kiveszi a legfelsőt és visszaadja az értékét. Ha nincs, akkor -1 a visszatérési érték.
- **Pop_big()**: ugyanez a nagyobb indexeken lévő veremre vonatkoztatva (itt a legkisebb index jelenti a verem tetejét).

6. Feladat, 5×1 pont. Válaszolj a következő kérdésekre röviden!

Igaz-e, hogy $2^n = O(n)$? Válaszod indokold!

Hamis, mert a 2^n függvény sokkal gyorsabban nő, mint az n . Az $f = O(g)$ pedig azt mondja, hogy az f függvény egy bizonyos pont után a g (konstansszoros) alatt kell végig maradjon, ami itt nem fordulhat elő.

A Huffman algoritmus minden lépésben két betűt/betűkapcsolatot olvaszt össze. Melyik kettőt?

Mindig a két legkisebb gyakoriságú betűt vagy betűkapcsolatot olvasztja össze.

Igaz-e, hogy a fa definíciója az, hogy körmentes gráf? Válaszod indokold!

Hamis. A fa teljes definíciójához hozzátartozik, hogy összefüggő is kell legyen.

Mi az a két metódus, amit Java-ban egyszerre kell felüldefiniálni, ha az osztályunk példányait hash táblában szeretnénk tárolni?

Az `equals()` és `hashCode()` metódusokat.

Egy tömb 400 ezer elemet tartalmaz. Az elemek intervallumáról nem tudunk semmit. Az egyenlő elemek egymáshoz viszonyított eredeti sorrendjét viszont szeretnénk megtartani. Melyik rendezőalgoritmust használnád? Válaszod indokold!

Stabil rendezést kell használjunk az egyforma elemek eredeti egymáshoz viszonyított sorrendjének megadásához. A kupac és a gyorsrendezés nem stabil. A leszámoló rendezést pedig nem használhatjuk, mert nem tudunk semmit a tartalmazott elemek intervallumáról. Az összefésülő rendezés viszont stabil, így azt érdemes használni.